N88-16362

Using Output to Evaluate and Refine Rules in Rule-Based Expert Systems*

D.C. St. Clair**, W.E. Bond, and B.B. Flachsbart
McDonnell Douglas Research Laboratories
McDonnell Douglas Corporation
St. Louis, MO

ABSTRACT

As space systems become increasingly complex and ambitious, the need for reliable expert systems to perform monitoring and diagnostic functions becomes more critical. Rule-based expert systems typically require large knowledge bases which must be carefully evaluated before being used in space vehicle operations. In the evaluation/refinement process, the knowledge engineer and domain experts evaluate expert system output and refine the rule base. The rule base size, coupled with rule interdependencies, makes this a very difficult task.

The research described suggests a method to compare the output set (E) of a rule-based expert system with a known set of correct conclusions (C) for a given set of input data and make decisions on how to refine the rule base. Using the techniques presented, system developers can evaluate and refine rules more accurately.

INTRODUCTION

Expert system evaluation/refinement attempts to insure that the conclusions of an expert system match those of a human expert. Typically, this process is accomplished by having the knowledge engineer input cases where behavior is known by the domain expert. The predictions made by the expert system are then compared to the "correct" answers. Where the results differ, the knowledge engineer works with domain experts to:

1. Locate and refine rules whose performance is questionable,
2. Identify missing rules and add them to the knowledge base,
3. Resolve conflicting rules, and
4. Remove extraneous rules.

This process is time-consuming and difficult to apply when the knowledge base is large and has many rule interdependencies. Learning techniques suggest some methods which can be used to expedite the process [1,3]. In particular, these techniques can be used to help isolate questionable rules and suggest avenues which should be explored to correct the problems.

The task is to modify a set of rules of the form hypothesis implies conclusion, viz.

$$H \rightarrow K$$

where H and K contain one or more propositions or negated propositions connected with conjunctions. The components of each rule come from a description space which has been determined by the knowledge engineer and the domain expert. Identification of the description space is an important and difficult problem since it contains all propositions used to describe the environment. Existing rules are refined by altering the propositions within H and/or K. New rules are created by combining propositions from the description space. The process of evaluation may indicate the description space needs to be expanded.

* This work was supported by the McDonnell Douglas Independent Research and Development program.

** Dr. St. Clair is Professor of Computer Science at the University of MO-Rolla, Graduate Engineering Center in St. Louis. He is currently on leave at MDRL.

The use of output to evaluate and refine rules necessitates the collection of some additional information. A trace of each rule chain producing system output along with corresponding rule unifications must be maintained for each test scenario to allow individual rules to be evaluated. In addition, two experience indicators must be maintained for each rule. A rule's "times fired" statistic is incremented each time it is fired. A rule's "times correct" statistic is incremented each time it participates in a rule chain leading to a correct system response. These statistics are used in the evaluation/refinement process.

COMPARISON OF EXPERT SYSTEM OUTPUT WITH KNOWN RESULTS

For a specific test scenario, three different conditions can be identified by comparing the output set (E) of the expert system with the set of known correct results (C). The three basic set relationships are shown in Figure 1.

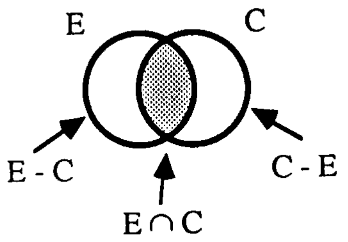

Figure 1. Comparison of Output with Known Results.

Each system output conclusion, $e_i \in E$, is the result of expert system input causing a chain of one or more rules to fire. Those $e_i \in E \cap C$ represent rule chains which terminate with a rule whose conclusion provides correct system response. Those $e_i \in E - C$ represent rule chains whose terminating rule conclusion produces an incorrect system response. Rule chains producing such output contain one or more incomplete or incorrect rules. Elements $c_i \in C - E$ indicate incorrect or missing rules.

The comparison of sets E and C for a specific test scenario will not identify cases where incorrect rule chains produce a correct conclusion. In many cases however, the identification of faulty rules in such chains will occur by applying the evaluation/refinement process to numerous test scenarios.

Some erroneous conditions can not be completely uncovered by comparing the contents of sets E and C. This includes cases where set E contains conflicting conclusions. In addition, each rule in an expert system which completely satisfies a suite of test scenarios will have identical "times fired" and "times correct" statistics. A rule whose "times fired" statistic is zero has not participated in the test suite.

EVALUATION/REFINEMENT OF RULES

The process of evaluating/refining rules consists not only of "fixing" incorrect or missing rules but of identifying rules which consistently work well and removing rules which are no longer needed by the system. The techniques described are iterative. To apply these techniques, each scenario in the test suite is processed by the expert system. Next, system evaluators use this output and the sets of known correct conclusions to perform the evaluation/refinement process. Then, the revised expert system reprocesses the test suite and the results are again evaluated/refined. Both the comprehensiveness of the test suite and the quality of expert system design determine the number of times this cycle must be repeated. During each cycle, rule experience indicators must be updated so they accurately reflect rule performance. This section suggests ways of performing the evaluation/refinement process.

Case 1: $e_i \in E - C$

The rule chain leading to a conclusion $e_i \in E - C$ represents an error of commission. It contains one or more rules which should not have fired. The evaluation/refinement of such rule chains requires solution of what Minsky termed the credit/blame assignment problem [4]. The solution of this problem identifies rules responsible for incorrect system behavior. Bundy, et al.[3] describe two basic techniques utilized by rule learning programs for identifying the first faulty rule within a chain. The identification of multiple faults within a chain requires repeated application of the evaluation/refinement process.

The first technique compares the actual rule chain with the chain which should have fired. Some programs require this ideal chain as input [2] while others [5] attempt to derive it by analysis using problem-solving and inference techniques. The first difference between the chains indicates which rule is faulty. The necessity of identifying the ideal rule chain makes this technique difficult to apply.

The second technique for finding a faulty rule is called Contradiction Backtracking. This technique, developed by Shapiro [7] does not require the identification of an ideal chain. Assuming the actual rule chain concludes with $e_i \in E - C$, Shapiro's algorithm begins by examining the last resolution step which lead to e_i. If the propositions which were resolved to produce e_i are true, select the branch of the tree which contains these propositions as part of the rule hypothesis, else select the other branch. Backtracking up the resolution tree is continued in this manner until a rule from the rule base is reached. This is the faulty rule. Both Shapiro and Bundy, et al. give examples of Contradiction Backtracking.

Once a suspect rule is located, its evaluation can lead to one of several conclusions.

1. The hypothesis of the rule is correct but the conclusion is incorrect. This situation is resolved by correcting the rule's conclusion.
2. The hypothesis of the rule is incorrect. Offending propositions in the hypothesis are replaced with correct ones.
3. The hypothesis of the rule is incomplete. Additional propositions must be added to the hypothesis to restrict the firing of the rule. The process of restricting a rule's application in this way is called discrimination [1]. The need for adding additional propositions to a hypothesis may lead to the discovery that the description space for the problem is incomplete.

The alteration of a rule should be done carefully since such changes are likely to effect other chains in which the rule participates. If the values of one rule's experience indicators vary from the experience indicators of other rules in the chain, it is highly likely at least one rule in the chain participates in other chains.

Case 2: $e_i \in E \cap C$

Since those conclusions $e_i \in E \cap C$ are correct, the "times correct" statistic is incremented for each rule in the associated chain. These statistics are helpful when trying to correct faulty rules, since they provide a history of each rule's performance. Incrementing these statistics indicate the rule has participated in a chain which leads to a correct conclusion. They do not indicate that each rule in the chain is correct.

Case 3: $c_i \in C - E$

A known correct conclusion $c_i \in C - E$ represents an error of omission. This can happen for two basic reasons:

1. A chain exists for producing this conclusion but it contains one or more incorrect rules, or
2. No chain resulting in this conclusion exists.

Finding existing faulty chains in this case is extremely difficult unless the ideal trace is known. In simple cases, it may be possible to find a rule whose conclusion matches c_i but whose hypothesis is incorrect. If a suspect rule can be found, its hypothesis may contain incorrect or overrestrictive propositions. In the latter case, it may be possible to generalize the rule by removing the overrestrictive propositions [1]. In many cases, generalization results in combining several rules into one. Since rules being generalized may participate in other rule chains, these chains must be examined before performing generalizations.

When no chain exists for producing a missing conclusion, one of two types of refinements may be made. A new chain can be created which terminates with a rule whose conclusion is c_i. In many cases, the basic system architecture helps suggest how the rule chain should be created. This process actually expands the rule base of the expert system. St. Clair, et al. [6] used this discovery technique in an adaptive diagnostic expert system which automatically refines its knowledge base. Alternately, it may be possible to add the missing conclusion to a rule which has produced a correct conclusion $e_i \in E \cap C$. This option is viable only if e_i and c_i always occur together. If this is not the case, a new rule chain should be added which terminates with the conclusion c_i.

Case 4: Other Cases for Evaluation/Refinement

Rule evaluation/refinement is incomplete as long as the system contains rules whose "times correct" to "times fired" statistics are not equal. Such rules are members of incorrect rule chains. If a rule is incorrect, it should be evaluated/refined as indicated above while carefully noting the chains in which it participates. Such a rule may make a correct contribution to some chains and an incorrect contribution to others. It may be necessary to replace rules of this type by one or more new rules.

A rule having a small "times fired" statistic has contributed very little to the expert system's operation. This may be due to the fact that the rule has not applied to the scenarios tested or it is extraneous and makes little or no contribution to system performance. The former case can be resolved by utilizing test scenarios which fire the rule. Extraneous rules occur as a result of errors in system design or because the refinement of other rules has removed them from rule chains. Removal of extraneous rules from the knowledge base may be desirable.

Set E must be reviewed to determine if conflicting conclusions exist. Conflicting conclusions result when one rule chain produces a conclusion inconsistent with that of another, for example, when one conclusion requests replacement of unit A and another requests adjustment of unit A. The rule chain belonging to the incorrect conclusion must be refined.

Cases in which $e_i = e_j$ for $i \neq j$, indicate that two or more rule chains led to the same conclusion. This condition may be a result of the original system design or it may arise from subsumption caused by use of the generalization and discrimination mechanisms described above. Repetition of results is not always undesirable; however, it serves as an indicator that the participating rule chains should be evaluated and those rules which are redundant should be removed.

CONCLUSIONS

The techniques described provide an effective tool which knowledge engineers and domain experts can utilize to help in evaluating and refining rules. These techniques have been used successfully as learning mechanisms in a prototype adaptive diagnostic expert system [6] and are applicable to other types of expert systems. The degree to which they constitute complete evaluation/refinement of an expert system depends on the thoroughness of their use.

REFERENCES

1. Blaxton T. A. and Kushner, B. G., *An Organizational Framework for Comparing Adaptive Artificial Intelligence Systems*, **1986 Proceedings of the Fall Joint Computer Conference,** IEEE Computer Society, November 1986, pp. 190-199.

2. Brazdil, P., *A Model for Error Detection and Correction*, Ph.D. Dissertation, University of Edinburgh, 1981.

3. Bundy, A., Silver, B., and Plummer, D., *An Analytical Comparison of Some Rule-Learning Programs*, **Artificial Intelligence**, Vol. 27, 1985, pp. 137-181.

4. Minsky, M., *Steps Towards Artificial Intelligence*, **Computers and Thought**, E.A. Feigenbaum and J. Feldman (Ed.s), New York: McGraw-Hill, 1963, pp. 406-450.

5. Mitchell, T.M., Utgoff, P.E., and Banerji, R., *Learning by Experimentation: Acquiring and Modifying Problem-Solving Heuristics*, **Machine Learning**, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Ed.s), Palo Alto, CA: Tioga, 1983, pp. 163-190.

6. St. Clair, D. C., Bond, W. E., Flachsbart, B. B., and Vigland, A. R., *An Architecture for Adaptive Learning in Rule-Based Diagnostic Expert Systems*, **1987 Proceedings of the Fall Joint Computer Conference**, IEEE Computer Society, October 1987.

7. Shapiro, E., *An Algorithm That Infers Theories From Facts*, **Proceedings of the Seventh International Joint Conference on Artificial Intelligence**, Los Altos, CA: William Kaufmann, Inc., 1981, pp. 446-451.